УДК 510.52

А.Н. Терещенко, В.К. Задирака
Институт кибернетики имени В.М. Глушкова НАН Украины, г. Киев
Украина, 03680, МСП, Киев-187, проспект Академика Глушкова, 40

Параллельный алгоритм вычисления циклической свертки

A.N. Tereshchenko, V.K. Zadiraka
V.M. Glushkov Institute of Cybernetics of NAS of Ukraine
Ukraine, 03680, Kyiv, 40 Glushkova ave

Parallel Computation Algorithm of Convolution

А.Н. Терещенко, В.К. Задірака
Інститут кібернетики імені В.М. Глушкова НАН України, м. Київ
Україна, 03680, МСП, Київ-187, проспект Академіка Глушкова, 40

Паралельний алгоритм обчислення циклічної згортки

Предложена оптимизация алгоритма вычисления циклической свертки длины $N = 2^n$ на основе быстрого преобразования Уолша (БПУ). Оптимизация заключается в использовании свойства предварительной корректировки сигнала длины M сигналом длины $M/2$ для получения суммарного результата БПУ двух сигналов на основе одного БПУ длины M . За счет этого уменьшается число БПУ более чем в два раза, что уменьшает сложность по числу операций сложения и вычитания. Показано, что алгоритм поддается распараллеливанию. Алгоритм проиллюстрирован на реализации сверток длины 4 и 8. Приведены оценки сложности при параллельном и последовательном вычислении.
Ключевые слова: многоразрядная арифметика, циклическая свертка, многоразрядное умножение.

It is given the optimization of computation algorithm for convolution of length $N = 2^n$ based on Fast Walsh transform (FWT). The optimization includes the use of the quality of pre-computational adjustment of multi-digit value of length M by multi-digit value of length $M/2$ to get the sum of FWTs both multi-digit values based on computation of one FWT multi-digit value of length M . The total number of FWTs is reduced more than twofold. That reduces the number of single precision additions and subtractions on sequential computation. It is shown that it is possible to compute algorithm in parallel. There are examples of computation algorithm of convolutions of length 4 и 8. Complexity of sequential and parallel computations is shown.
Key Words: multidigit arithmetic, cyclic convolution, multidigit multiplication.

Запропонована оптимізація алгоритму обчислення циклічної згортки довжини $N = 2^n$ на основі швидкого перетворення Уолша (ШПУ). Оптимізація полягає у використанні властивості попереднього корегування сигналу довжини M сигналом довжини $M/2$ для отримання сумарного результату ШПУ двох сигналів на основі одного ШПУ довжини M . За рахунок цього зменшується число ШПУ більш ніж у два рази, що зменшує складність за кількістю операцій додавання та віднімання при послідовному обчисленні. Показано, що алгоритм піддається розпаралелюванню. Алгоритм проілюстровано на реалізації згорток довжини 4 и 8. Наведені оцінки складності при послідовному та паралельному обчисленні.
Ключові слова: багаторозрядна арифметика, циклічна згортка, багаторозрядне множення.

Введение

Аппаратная часть совершенствуется с каждым годом. Применяются новые технологии и методы, которые повышают производительность. Одновременно с повышением быстродействия повышаются требования ко времени выполнения алгоритмов шиф-

рования-дешифрования, которые оперируют с более длинными ключами (не менее 1024 бит). Быстродействие операции умножения является актуальной задачей, так как от быстродействия данной операции в основном зависит время выполнения операций в криптоалгоритмах.

Операция быстрого умножения может быть реализована с помощью алгоритмов вычисления свертки. Данная работа является продолжением работ [1-5] по вычислению циклической свертки **с целью** *максимального распараллеливания всех шагов алгоритма и повышения эффективности алгоритма*.

Постановка задачи. На основе БПУ вычислить циклическую свертку R_N разрядностью N, где $N = 2^n$, $n > 0$ – целые, двух последовательностей X_N и Y_N, используя следующее выражение $R_N = X_N \otimes Y_N$, $R_N = (r_0 ... r_{N-1})$, $r_k = \sum_{p=0}^{N-1} x_p y_{\langle p+k \rangle_N}$, $k = \overline{0, N-1}$, при последовательном выполнении на одном процессоре и при параллельном выполнении на P процессорах (графического ускорителя).

Обозначения.

Последовательности длины N могут быть представлены в виде векторов:

$$X_N = \begin{bmatrix} x_0 \\ x_1 \\ \\ x_{N-1} \end{bmatrix}, Y_N = \begin{bmatrix} y_0 \\ y_1 \\ \\ y_{N-1} \end{bmatrix}.$$

Циклическую свертку разрядностью $N = 4$ можно представить в виде:

$$\begin{array}{c|cccc} x_0 & y_0 & y_1 & y_2 & y_3 \\ x_1 & y_1 & y_2 & y_3 & y_0 \\ x_2 & y_2 & y_3 & y_0 & y_1 \\ x_3 & y_3 & y_0 & y_1 & y_2 \\ \hline & r_0 & r_1 & r_2 & r_3 \end{array} \quad \text{или} \quad \begin{array}{c|cccc} y_0 & x_0 & x_1 & x_2 & x_3 \\ y_1 & x_1 & x_2 & x_3 & x_0 \\ y_2 & x_2 & x_3 & x_0 & x_1 \\ y_3 & x_3 & x_0 & x_1 & x_2 \\ \hline & r_0 & r_1 & r_2 & r_3 \end{array}.$$

Далее будет использоваться именно это представление.

$X = (X, Z)$ обозначает, что к элементам X добавляются элементы Z справа. Соответственно длина X увеличивается на число элементов Z.

$((X_M)_i)$ обозначает, что в X рассматривается i-я секция. Считается, что X разбит на секции длины $M = 2^d$, где $d > 0$ – целое.

x_i обозначает i-й элемент X. При этом вектор X считается полнозаполненным без учета мнимой разбивки на секции разрядностью степени двойки.

В выражениях последовательность операций будем разбивать на группы символом ‘;‘. Группы выполняются в обычном порядке слева направо. Операции внутри группы разделяются запятой и выполняются в обратном порядке – справа налево. Если операцию обозначить номером ее выполнения, то получим строку, которая определяет порядок выполнения операций:

2, 1; 6, 5, 4, 3; 8, 7.

Введем следующие операторы: E, O, U, L, H, S, A, W, B ($E - Even$ (четный), $O - Odd$ (нечетный), $U - Up$ (вверх), $L - Low$ (младший), $H - High$ (старший),

$S - Substruct$ (вычитание), $A - Addition$ (добавление), $W - Walsh$ (одна итерация преобразования Уолша), $B - Bit\ inverse\ order$ (бит-инверсная перестановка) и обозначим их следующим образом:

$$\left[E(X_N)\right]_k = x_{2k},\ \left[O(X_N)\right]_k = x_{2k+1},\ \left[L(X_N)\right]_k = x_k,\ \left[H(X_N)\right]_k = x_{k+N/2},\ k = \overline{0, N/2-1};$$

циклический сдвиг элементов вверх – $V_N = U(X_N)$, $v_k = x_{\langle k+1\rangle_N}$, $k = \overline{0, N-1}$;

$$S(Z_N) = E(Z_N) - O(Z_N),\ A(Z_N) = E(Z_N) + O(Z_N);$$

$$Z_N \leftarrow W(X_N),\ \begin{Bmatrix} E(Z_N) \leftarrow A(X_N) = E(X_N) + O(X_N) \\ O(Z_N) \leftarrow S(X_N) = E(X_N) - O(X_N) \end{Bmatrix}.$$

Проиллюстрируем введенные операторы для случаев $N = 4, 8$:

$$E(X_8) = \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix},\ O(X_8) = \begin{bmatrix} x_1 \\ x_3 \\ x_5 \\ x_7 \end{bmatrix},\ U(Y_4) = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_0 \end{bmatrix},\ L(X_8) = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix},\ H(X_8) = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix},$$

$$S(X_8) = \begin{bmatrix} x_0 - x_1 \\ x_2 - x_3 \\ x_4 - x_5 \\ x_6 - x_6 \end{bmatrix},\ A(Y_8) = \begin{bmatrix} y_0 + y_1 \\ y_2 + y_3 \\ y_4 + y_5 \\ y_6 + y_6 \end{bmatrix},\ W(X_4) = \begin{bmatrix} x_0 - x_1 \\ x_0 + x_1 \\ x_2 + x_3 \\ x_2 - x_3 \end{bmatrix},\ B(X_4) = \begin{bmatrix} x_0 \\ x_2 \\ x_1 \\ x_3 \end{bmatrix}.$$

Матрицы дискретного преобразования Уолша (с упорядочиванием по Пели), которые необходимо отличать от оператора W, обозначим следующим образом:

$$W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix},\ W_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}.$$

Лемма 1. При вычислении БПУ сигнала X_N, где $N = 2^n$, $n > 0$ – целые, оператор W длины N выполняется n раз над вектором X_N.

Доказательство. Вычисление БПУ можно организовать следующим образом:

$$((X_M)_j) \leftarrow \left[\frac{B(L((X_M)_j))}{B(H((X_M)_j))}\right],\ (X_M)_j) \leftarrow B(W((X_M)_j)),\ j = \overline{0, 2^i - 1},\ M = 2^{n-i},$$

$$i = \overline{0, n-1}.$$

Вычисление БПУ длины $N = 2^2 = 4$ согласно предыдущей формуле имеет вид:

$$\hat{X}_4 \leftarrow \begin{bmatrix} B(L(L(\hat{X}_4))) \\ \hdashline B(H(L(\hat{X}_4))) \\ \hdashline B(L(H(\hat{X}_4))) \\ \hdashline B(H(H(\hat{X}_4))) \end{bmatrix},\ \hat{X}_4 \leftarrow \begin{bmatrix} B(W(L(\hat{X}_4))) \\ \hline B(W(H(\hat{X}_4))) \end{bmatrix},\ \hat{X}_4 \leftarrow \begin{bmatrix} B(L(\hat{X}_4)) \\ \hline B(H(\hat{X}_4)) \end{bmatrix},\ \hat{X}_4 \leftarrow B(W(X_4)).$$

Так как оператор B (бит-инверсная перестановка) не переставляет элементы, если длина вектора равна 1 или 2, то оператор B и сопутствующие ему операторы L, H могут быть опущены в предыдущих выражениях, кроме $\hat{X}_4 \leftarrow B(W(X_4))$. Так как $W(Z_4) = \begin{bmatrix} W(L(Z_4)) \\ \hline W(H(Z_4)) \end{bmatrix}$, то окончательное выражение для вычисления БПУ длины 4 будет $\hat{X}_4 \leftarrow W(B(W(X_4)))$, где W применяется по всей длине вектора.

Более детально данное преобразование (справа налево) представлено ниже:

$$\hat{X}_4 = \begin{bmatrix} x_0+x_1+x_2+x_3 \\ x_0+x_1-x_2-x_3 \\ x_0-x_1+x_2-x_3 \\ x_0-x_1-x_2+x_3 \end{bmatrix} \leftarrow W\left(\begin{bmatrix} x_0+x_1 \\ x_2+x_3 \\ x_0-x_1 \\ x_2-x_3 \end{bmatrix}\right), \begin{bmatrix} x_0+x_1 \\ x_2+x_3 \\ x_0-x_1 \\ x_2-x_3 \end{bmatrix} \leftarrow B\left(\begin{bmatrix} x_0+x_1 \\ x_0-x_1 \\ x_2+x_3 \\ x_2-x_3 \end{bmatrix}\right),$$

$$\begin{bmatrix} x_0+x_1 \\ x_0-x_1 \\ x_2+x_3 \\ x_2-x_3 \end{bmatrix} \leftarrow W\left(X_4 = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}\right).$$

В БПУ длины $N = 8$ оператор W также можно применять по всей длине:

$$\hat{X}_8 \leftarrow W\left(\begin{bmatrix} B(L(\hat{X}_8)) \\ \hdashline B(H(\hat{X}_8)) \end{bmatrix}\right),\ \hat{X}_8 \leftarrow W\left(\begin{bmatrix} B(L(\hat{X}_8)) \\ \hdashline B(H(\hat{X}_8)) \end{bmatrix}\right),\ \hat{X}_8 \leftarrow B(W(X_8)).$$

В работе [6] используется алгоритм с высоким уровнем распараллеливания и сохранением структуры графа преобразования Уолша с различными системами упорядочивания по Пели, Уолшу – Качмажу и преобразование Уолша с четным – нечетным упорядочиванием, которые требуют меньшее число перестановок для ускорения распараллеливания.

Вычисление БПУ длины $N = 16$ будет следующим:

$$\hat{X}_{16} \leftarrow W\left(\begin{bmatrix} B(L(L(\hat{X}_{16}))) \\ \hdashline B(H(L(\hat{X}_{16}))) \\ \hdashline B(L(H(\hat{X}_{16}))) \\ \hdashline B(H(H(\hat{X}_{16}))) \end{bmatrix}\right),\ \hat{X}_{16} \leftarrow W\left(\begin{bmatrix} B(L(L(\hat{X}_{16}))) \\ \hdashline B(H(L(\hat{X}_{16}))) \\ \hdashline B(L(H(\hat{X}_{16}))) \\ \hdashline B(H(H(\hat{X}_{16}))) \end{bmatrix}\right),$$

$$\hat{X}_{16} \leftarrow \begin{bmatrix} B(L(\hat{X}_{16})) \\ \hdashline B(H(\hat{X}_{16})) \end{bmatrix},\ \hat{X}_{16} \leftarrow W\left(\begin{bmatrix} B(L(\hat{X}_{16})) \\ \hdashline B(H(\hat{X}_{16})) \end{bmatrix}\right),\ \hat{X}_{16} \leftarrow B(W(X_{16})).$$

Продолжая по индукции, можно показать, что, применяя B, L и H, можно организовать вычисление таким образом, что W на каждой итерации применяется над всеми элементами. Лемма доказана.

Следствие. При параллельном вычислении БПУ длины $N = 2^n$, где $n > 0$ – целое, каждый из N процессоров выполнит только n операций сложения (вычитания).

Перед тем как перейти к описанию нового алгоритма, рассмотрим следующее свойство сложения БПУ сигналов разной разрядности.

Лемма 2. Если сигналы $Z_M = \sum_{k=0}^{M-1} z_k \cdot 2^{\omega}$, $V_{M/2} = \sum_{k=0}^{M/2-1} v_k \cdot 2^{\omega}$, где ω – длина слова в битах, $M = 2^p$, $p > 0$, формируют сигнал $R_M = \sum_{k=0}^{M-1} r_k \cdot 2^{\omega}$ вида: $r_s = 2z_s + v_k$, $r_{s+t} = 2z_{s+t} - v_k$, $s = k + t \cdot \left\lfloor \frac{k}{t} \right\rfloor$, $k = \overline{0, M/2-1}$, где $t = \frac{M}{2q}$, $t = 2^m$, $1 \le t, q \le \frac{M}{2}$, $m > 0$, то БПУ $\hat{Z}_M = \sum_{k=0}^{M-1} \hat{z}_k \cdot 2^{\omega}$, $\hat{V}_{M/2} = \sum_{k=0}^{M/2-1} \hat{v}_k \cdot 2^{\omega}$, $\hat{R}_M = \sum_{k=0}^{M-1} \hat{r}_k \cdot 2^{\omega}$ связаны следующими выражениями:

$$\hat{r}_d = 2\hat{z}_d + 2\hat{v}_k,\ \hat{r}_{d-q} = 2\hat{z}_{d-q},\ d = k + q \cdot \left\lfloor 1 + \frac{i}{q} \right\rfloor,\ k = \overline{0, M/2-1}.$$ [1]

Доказательство. Заметим, что выражения $s = k + t \cdot \lfloor k/t \rfloor$, $d = k + q \cdot \lfloor 1 + k/q \rfloor$ имеют одинаковую структуру, с учетом того, что t и q взаимосвязаны $\left(t = \frac{M}{2q} \right)$ и определяют длину последовательности значений идущих по порядку (табл. 1).

Таблица 1 – Зависимость значений s и d от значений t, q, M

		$k =$	0	1	2	3	4	5	6	7
$M = 8$	$t = 4$	s	0	1	2	3				
	$q = 1$	d	1	3	5	7				
$M = 8$	$t = 2$	s	0	1	4	5				
	$q = 2$	d	2	3	6	7				
$M = 8$	$t = 1$	s	0	2	4	6				
	$q = 4$	d	4	5	6	7				
$M = 16$	$t = 8$	s	0	1	2	3	4	5	6	7
	$q = 1$	d	1	3	5	7	9	11	13	15
$M = 16$	$t = 4$	s	0	1	2	3	8	9	10	11
	$q = 2$	d	2	3	6	7	10	11	14	15
$M = 16$	$t = 2$	s	0	1	4	5	8	9	12	13
	$q = 4$	d	4	5	6	7	12	13	14	15
$M = 16$	$t = 1$	s	0	2	4	6	8	10	12	14
	$q = 8$	d	8	9	10	11	12	13	14	15

Рассмотрим результаты сложения БПУ сигналов $Z_8 = (0, 0, 0, 0, 0, 0, 0, 0)$ и $V_4 = (1, 4, 9, 25)$. Найдем сначала БПУ сигнала V_4.

$$\hat{V}_4 = W_4 \cdot V_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 4 \\ 9 \\ 25 \end{bmatrix} = \begin{bmatrix} 39 \\ -29 \\ -19 \\ 13 \end{bmatrix}.$$

Рассмотрим результаты сложения при $q = 1, 2, 4$.[2] Получим результат:

$$\hat{R}_8^{t=4,q=1} = W_8 \cdot \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ -v_0 \\ -v_1 \\ -v_2 \\ -v_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 4 \\ 9 \\ 25 \\ -1 \\ -4 \\ -9 \\ -25 \end{bmatrix} = \begin{bmatrix} 0 \\ 78 \\ 0 \\ -58 \\ 0 \\ -38 \\ 0 \\ 26 \end{bmatrix} = 2 \cdot \begin{bmatrix} 0 \\ \hat{d}_0 \\ 0 \\ \hat{d}_1 \\ 0 \\ \hat{d}_2 \\ 0 \\ \hat{d}_3 \end{bmatrix}.$$

$$\hat{R}_8^{t=2,q=2} = W_8 \cdot \begin{bmatrix} v_0 \\ v_1 \\ -v_0 \\ -v_1 \\ v_2 \\ v_3 \\ -v_2 \\ -v_3 \end{bmatrix} = W_8 \cdot \begin{bmatrix} 1 \\ 4 \\ -1 \\ -4 \\ 9 \\ 25 \\ -9 \\ -25 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 78 \\ -58 \\ 0 \\ 0 \\ -38 \\ 26 \end{bmatrix} = 2 \cdot \begin{bmatrix} 0 \\ 0 \\ \hat{v}_0 \\ \hat{v}_1 \\ 0 \\ 0 \\ \hat{v}_2 \\ \hat{v}_3 \end{bmatrix}, \quad \hat{R}_8^{t=1,q=4} = W_8 \cdot \begin{bmatrix} v_0 \\ -v_0 \\ v_1 \\ -v_1 \\ v_2 \\ -v_2 \\ v_3 \\ -v_3 \end{bmatrix} = W_8 \cdot \begin{bmatrix} 1 \\ -1 \\ 4 \\ -4 \\ 9 \\ -9 \\ 25 \\ -25 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 78 \\ -58 \\ -38 \\ 26 \end{bmatrix} = 2 \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \hat{v}_0 \\ \hat{v}_1 \\ \hat{v}_2 \\ \hat{v}_3 \end{bmatrix}.$$

Изучив исходные данные и результат выражений $\hat{R}_8^{q=1}$, $\hat{R}_8^{q=2}$, $\hat{R}_8^{q=4}$, видим, что, прибавив определенным образом элементы сигнала V_4 к элементам сигнала Z_8, одновременно с вычислением преобразования Уолша сигнала $\hat{Z}_8$ вычисляется преобразование сигнала $\hat{V}_4$. Необходимо также использовать поправочный коэффициент 2 для Z_8, так как $\hat{R}_8^{q=1}$, $\hat{R}_8^{q=2}$, $\hat{R}_8^{q=4}$ содержат удвоенное значение $\hat{V}_4$.

Указанное в лемме свойство сохраняется при больших разрядностях свертки, рассматривая по индукции. Лемма доказана.

Примечание 1. Другими словами, q – длина секции $\hat{R}_M$, к которой последовательно добавляются элементы из $\hat{V}_{M/2}$.

Примечание 2. В представленном алгоритме используется только случай при $q = 1$ (см. также Шаг 9б алгоритма).

Вычисление циклической свертки разрядностью $N = 4$ *на основе БПУ*. В работе [3] приведен метод вычисления циклической свертки с использованием БПУ, основанный на вычислении сверток половинной длины. Алгоритм [3] осуществляет разбиение до сверток длины $N = 4$, для вычисления которой достаточно 5 операций однословного умножения. Далее приведен новый алгоритм вычисления свертки на основе БПУ

длины $N = 4$, при котором обратное БПУ выполняется в качестве последней операции, что гораздо удобнее при реализации параллельного алгоритма:

$$R_4 \leftarrow \frac{1}{8} \cdot W_4 \cdot \hat{A}_4 ,\ \hat{A}_4 \leftarrow 2 \cdot \hat{X}_4 \cdot \hat{Y}_4 + T_4 ,\ T_4 \leftarrow \begin{bmatrix} 0 \\ -t \\ 0 \\ +t \end{bmatrix},$$

$$t \leftarrow S(O(\hat{X}_4)) \cdot A(O(\hat{Y}_4)) ,\ \begin{matrix} \hat{X}_4 \leftarrow W_4 \cdot X_4 \\ \hat{Y}_4 \leftarrow W_4 \cdot Y_4 \end{matrix},$$

где X_4, Y_4 – входные сигналы; R_4 – результат вычисления циклической свертки сигналов X_4 и Y_4; $t = (x_1 - x_3) \cdot (y_0 - y_2) = x_1 y_0 + x_3 y_2 - x_1 y_2 - x_3 y_0$.

Использование операторов S, A над векторами $\hat{X}_4$, $\hat{Y}_4$ для вычисления t является резервом оптимизации, так как дает возможность не сохранять X_4 и Y_4.

Алгоритм 1. Реализация операции циклической свертки двух последовательностей длины $N = 2^n$ с использованием БПУ и параллельных вычислений.

Параметры: X_N, Y_N – последовательности длины $N = 2^n$, где $n \geq 3$, n – целое.

Результат: $R_N \leftarrow X_N \otimes Y_N$ – циклическая свертка сигналов X_N, Y_N.

Шаг 0. Инициализация. $X \leftarrow ((X_{N=2^n})_0) \leftarrow X_N$; $Y \leftarrow ((Y_{N=2^n})_0) \leftarrow Y_N$.

Шаг 1. БПУ начальной секции $((X_N)_0)$. $((X_N)_0) \leftarrow W_N \cdot ((X_N)_0)$.

Шаг 2. ДПУ остальных секций вектора X как линейные комбинации секций X.

$$X \leftarrow (X, V_{M/2}) ,\ V_{M/2} \leftarrow L((X_M)_j) - H((X_M)_j) ,$$

$$j = \overline{0, T-1} ,\ M = 2^{n-i} ,\ T = 3^i ,\ i = \overline{0, n-3} .$$ [3,4]

Шаг 3. БПУ секций вектора Y (за исключением двух последних итераций).

Шаг 3а. Для i с 0 по $n-3$

Шаг 3б. $Y \leftarrow (Y, V_{M/2}) ,\ V_{M/2} \leftarrow U(E(Z_M)) - E(Z_M) ,$

Шаг 3в. $((Y_M)_j) \leftarrow \begin{bmatrix} \dfrac{B(L((Y_M)_j))}{B(H((Y_M)_j))} \end{bmatrix} ,\ ((Y_M)_j) \leftarrow B(W(Z_M)) ,$

Шаг 3г. $Z_M \leftarrow ((Y_M)_j) ,\ j = \overline{0, T-1} ,\ M = 2^{n-i} ,\ T = 3^i .$

Шаг 3д. Конец цикла по i. [3,4]

Шаг 4. Последние 2 итерации БПУ. $((Y_4)_j) \leftarrow W(B(W((Y_4)_j)))$, $j = \overline{0, 3^{n-2} - 1}$ (см. Лемму 1).

Шаг 5. Дополнительные секции

$$XADD_{3^{n-2}} \leftarrow S(O(X_{4 \cdot 3^{n-2}})) ,\ YADD_{3^{n-2}} \leftarrow A(O(Y_{4 \cdot 3^{n-2}})) .$$

Шаг 6. Поэлементное умножение основных секций. $R_{4 \cdot 3^{n-2}} \leftarrow 2 \cdot X_{4 \cdot 3^{n-2}} \cdot Y_{4 \cdot 3^{n-2}}$.

Шаг 7. Поэлементное умножение дополнительных секций.

$$RADD_{3^{n-2}} \leftarrow XADD_{3^{n-2}} \cdot YADD_{3^{n-2}} .$$

Шаг 8. Подготовка к вычислению сверток длины 4.

$$r_{4j+1} \leftarrow r_{4j+1} - radd_j ,\ r_{4j+3} \leftarrow r_{4j+3} + radd_j ,\ j = \overline{0, 3^{n-2} - 1} .$$ [4]

Шаг 9. Подготовка к вычислению сверток длины 8 и больше.

Шаг 9а. Для i с $n-3$ до 0 шаг –1.

Шаг 9б. $$((R_M)_j) \leftarrow \left[\frac{L(Z_M)+V_{M/2}}{H(Z_M)-V_{M/2}}\right],\ Z_M \leftarrow ((R_M)_j),$$

$$V_{M/2} \leftarrow \frac{1}{2}\cdot((R_{M/2})_{2T+j}),\ j=\overline{0,T-1},\ M=2^{n-i},\ T=3^i.^4$$

Шаг 9в. Конец цикла по i.

Шаг 10. Вычисление ОБПУ. $R_N \leftarrow \frac{1}{N}\cdot W_N \cdot \frac{1}{2}\cdot((R_N)_0)$.

Примечание 3. Длина X (или Y) после каждой итерации по i увеличивается и равна $2^n\cdot\left(\frac{3}{2}\right)^{i+1}$. По завершении цикла по i ($i=n-3$) длина X (или Y) равна $4\cdot 3^{n-2}$ (см. Лемма 4 и Теорема 1).

Примечание 4. В алгоритме i соответствует номеру итерации, j – номеру обрабатываемой секции на итерации i. При обработке Y мнимо разбивается на секции одинаковой длины $M=2^{n-i}$. При вычислении X и R на итерации i рассматриваются секции длины M и $M/2$.

Ниже приведена схема вычисления свертки длины $N=8$ на основе приведенного алгоритма.

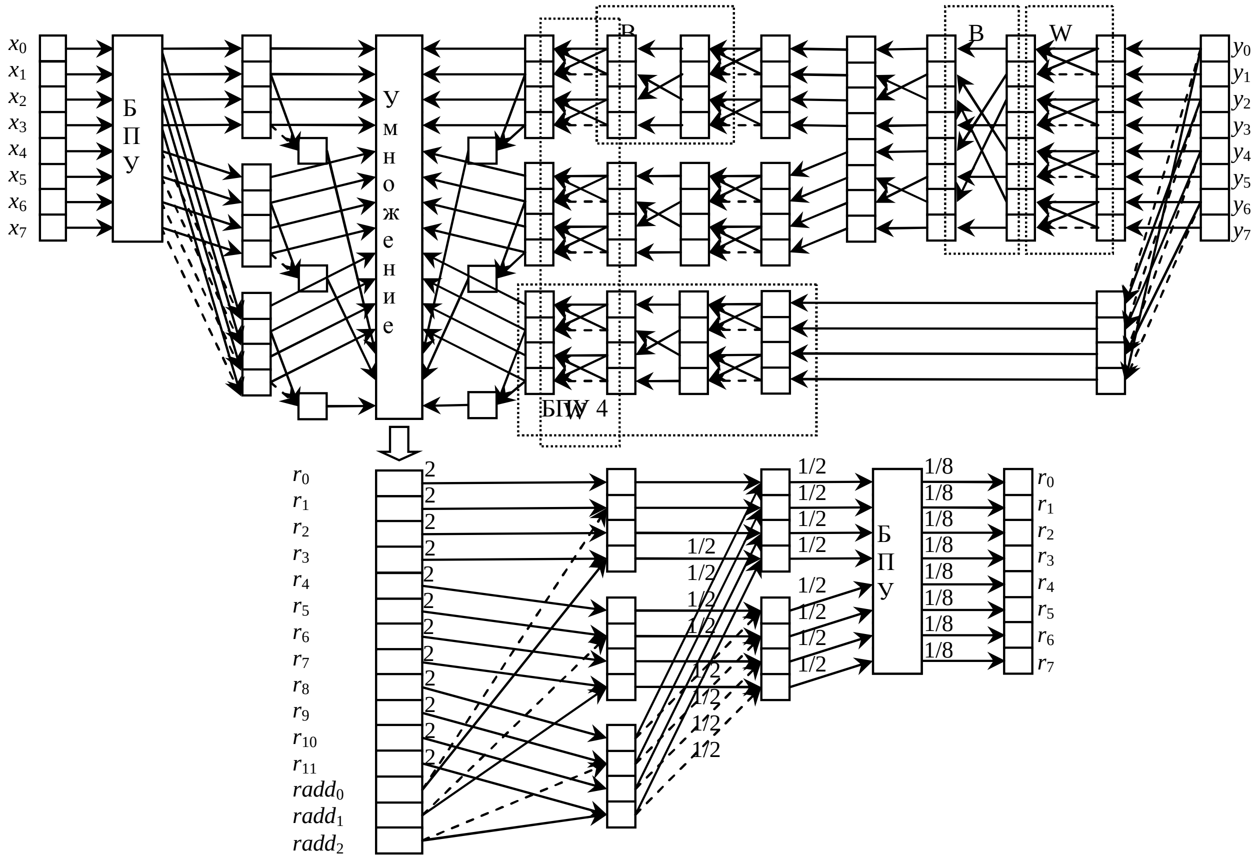


Рисунок 1 – Схема вычисления свертки длины $N=8$ на основе Алгоритма 1

Число операций при последовательном выполнении алгоритма.

Лемма 3. При последовательном вычислении свертки сложность по числу однословных операций умножения равна $5\cdot 3^{n-2}$.

Доказательство. Согласно Шагам 6 и 7 Алгоритма 1 число элементов основных и дополнительных секций при поэлементном умножении равно $4 \cdot 3^{n-2}$ и 3^{n-2} соответственно. Лемма доказана.

Лемма 4. При последовательном вычислении свертки сложность по числу однословных операций сложения (вычитания) равна $5 \cdot 3^{n-2} + (n-1) \cdot 2^n$ для нахождения элементов в основных $X_{4 \cdot 3^{n-2}}$ и дополнительных $XADD_{3^{n-2}}$ секциях.

Доказательство. В табл. 2 показана зависимость числа основных секций от длины N входной последовательности и определено число итераций, необходимых для построения основных секций X.

Таблица 2 – Итерационное вычисление вектора длин основных секций X при вычислении сверток длины $N = 2^n$, $n = 3, 4, 5, 6$ (Шаг 2)

i	$N = 8, n = 3$
0	8, 4
i	$N = 16, n = 4$
0	16, 8
1	8,8,8, 4,4,4
i	$N = 32, n = 5$
0	32,16
1	16,16,16, 8,8,8
2	8,8,8, 8,8,8, 8,8,8, 4,4,4, 4,4,4, 4,4,4

i	$N = 64, n = 6$
0	64,32
1	32,32,32, 16,16,16
2	16,16,16, 16,16,16, 16,16,16, 8,8,8, 8,8,8, 8,8,8
3	8,8,8, 8,8,8, 8,8,8, 8,8,8, 8,8,8, 8,8,8, 8,8,8, 8,8,8, 8,8,8, 4,4,4, 4,4,4, 4,4,4, 4,4,4, 4,4,4, 4,4,4, 4,4,4, 4,4,4, 4,4,4

Из табл. 2 видно, что на каждой итерации число элементов увеличивается в $3/2$ раза, так как вычисление каждой свертки длины M (согласно методу Питасси [1]) основано на трех свертках половинной длины $M/2$. Число элементов X на последней итерации $n-3$ равно:

$$2^n \cdot \left(\frac{3}{2}\right)^{(n-3)+1} = 4 \cdot 3^{n-2}.$$

Алгоритм БПУ используется только для вычисления ДПУ начальной основной секции X длины $N = 2^n$, для чего необходимо $n \cdot N$ одноразрядных операций сложения (вычитания). ДПУ всех остальных $X_{4 \cdot 3^{n-2}}$ и $XADD_{3^{n-2}}$ секций вычисляются как линейные комбинации ДПУ начальной основной секции X на Шагах 2 и 5 с использованием операторов L, H, S и O, что является резервом оптимизации и позволяет уменьшить число вычислений ДПУ на треть, что существенно для длинных сверток.

С учетом числа операций для вычисления БПУ начальной секции длины N общее число одноразрядных операций сложения (вычитания) для вычисления секций в $X_{4 \cdot 3^{n-2}}$ (Шаги 1 и 2) будет равно $4 \cdot 3^{n-2} - N + n \cdot N = 4 \cdot 3^{n-2} + (n-1) \cdot 2^n$, где $N = 2^n$, $n > 0$ – целое. При применении операторов S и O (каждый из которых уменьшает $X_{4 \cdot 3^{n-2}}$ в два раза) на Шаге 5 для вычисления секций $XADD_{3^{n-2}}$ необходимо 3^{n-2} операции вычитания.[5] Таким образом, общее число одноразрядных операций сложения (вычитания) составляет $4 \cdot 3^{n-2} + (n-1) \cdot 2^n + 3^{n-2} = 5 \cdot 3^{n-2} + (n-1) \cdot 2^n$.

Лемма доказана.

Примечание 5. В отличие от алгоритма [3] в предложенном алгоритме БПУ в дополнительных секциях ($XADD_{3^{n-2}}$ и $YADD_{3^{n-2}}$) не выполняется, что также является резервом оптимизации и значительно уменьшает общее число операций сложения (вычитания).

Теорема 1. При последовательном вычислении свертки сложность по числу однословных операций сложения (вычитания) равна $13 \cdot 3^{n-2} - 3 \cdot 2^n$ для нахождения основных $Y_{4 \cdot 3^{n-2}}$ и дополнительных $YADD_{3^{n-2}}$ секций.

Доказательство. Особенностью вычисления элементов Y является то, что на Шаге 3б используется циклический сдвиг $U(E(Z_M)) - E(Z_M)$, который не может быть получен как результат линейных комбинаций ДПУ секций Y. Поэтому в Y необходимо выполнять БПУ в каждой секции полностью на Шаге 3в. Вычисление алгоритма можно организовать таким образом, что на каждой итерации $i = \overline{0, n-3}$ для каждой секции Y выполняется только один шаг БПУ $((Y_M)_j) \leftarrow \begin{bmatrix} B(L((Y_M)_j)) \\ B(H((Y_M)_j)) \end{bmatrix}$, $((Y_M)_j) \leftarrow B(W((Y_M)_j))$ (см. Лемму 1) за исключением секций длины 4.

В табл. 3 представлено по секциям число операций сложения (вычитания), необходимых для завершения одного шага БПУ. На каждой итерации i оператор W применяется для секций одинаковой длины $M = 2^{n-i}$. Разбиение на секции необходимо, так как после применения оператора W применяется оператор B (бит-инверсная сортировка), результат которого зависит от длины секции $M = 2^{n-i}$.

Таблица 3 – Зависимость числа операций сложения (вычитания) одного шага БПУ для каждой секции Y длины $M = 2^{n-i}$, $i = \overline{0, n-3}$, при вычислении сверток длины $N = 2^n$, $n = 3, 4, 5, 6$ (Шаги 3в и 4)

i	Шаг №	$N = 8, n = 3$
0	Шаг 3в	8
	Шаг 4	4,4,4
	Шаг 4	4,4,4
i		$N = 16, n = 4$
0	Шаг 3в	16
1	Шаг 3в	8,8,8
	Шаг 4	4,4,4,4,4,4,4,4,4
	Шаг 4	4,4,4,4,4,4,4,4,4
i		$N = 32, n = 5$
0	Шаг 3в	32
1	Шаг 3в	16,16,16
2	Шаг 3в	8,8,8, 8,8,8, 8,8,8
	Шаг 4	4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4.[6]
	Шаг 4	4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4.[6]

i	Шаг №	$N = 64, n = 6$
0	Шаг 3в	64
1	Шаг 3в	32,32,32
2	Шаг 3в	16,16,16, 16,16,16, 16,16,16
3	Шаг 3в	8,8,8, 8,8,8, 8,8,8, 8,8,8, 8,8,8, 8,8,8, 8,8,8, 8,8,8, 8,8,8
	Шаг 4	4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4.[5]
	Шаг 4	4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4,4,4,4, 4,4,4.[6]

Примечание 6. Пробелы добавлены для удобства восприятия.

Из табл. 3 видно, что на последующей итерации (Шаг 3в) длина секций уменьшается в два раза и число секций увеличивается в три раза, следуя методу Питасси.

Общее число операций сложения (вычитания), необходимых для пошагового вычисления БПУ (только Шаг 3в) в каждой секции, может быть получено как сумма геометрической прогрессии по следующей формуле:

$$Q^{\pm}_{Шаг\ 3в}(Y_T)_i = 2^n \cdot \left(\frac{1-q^{i+1}}{1-q}\right),\ T = 2^n \cdot q^{i+1},\ i = \overline{0, n-3},\ q = \frac{3}{2},$$

где $Q^{\pm}_{Шаг\ 3в}(Y_T)_i$ – сумма числа операций сложения (вычитания) над элементами Y_T по i-ю итерацию включительно.

С учетом предыдущей формулы общее число операций сложения (вычитания) над элементами Y на Шаге 3в по завершении итерации $i = n-3$ может быть выражено как $Q^{\pm}_{Шаг\ 3в}(Y_{4\cdot 3^{n-2}}) = 8\cdot 3^{n-2} - 2^{n+1}$ при $n \geq 2$.

На Шаге 3б длина Y увеличивается на каждой итерации по j и по i. Тогда для нахождения остальных элементов (кроме исходной начальной секции длины $N = 2^n$) необходимо $4\cdot 3^{n-2} - 2^n$ операций по аналогии с X. По завершении Шагов 2 и 3а-3д длины $X_{4\cdot 3^{n-2}}$ и $Y_{4\cdot 3^{n-2}}$ равны. С учетом 3^{n-2} операций сложения для нахождения $YADD_{3^{n-2}}$ на Шаге 4 общее число операций сложения (вычитания) основных $Y_{4\cdot 3^{n-2}}$ и дополнительных $YADD_{3^{n-2}}$ секций[5] будет следующим:

$$Q^{\pm}(Y_{4\cdot 3^{n-2}}, YADD_{3^{n-2}}) = Q^{\pm}_{Шаг\ 3б}(Y_{4\cdot 3^{n-2}}) + Q^{\pm}_{Шаг\ 3в}(Y_{4\cdot 3^{n-2}}) + Q^{\pm}_{Шаг\ 4}(YADD_{3^{n-2}}).$$

$$Q^{\pm}(Y_{4\cdot 3^{n-2}}, YADD_{3^{n-2}}) = \left(4\cdot 3^{n-2} - 2^n\right) + \left(8\cdot 3^{n-2} - 2^{n+1}\right) + 3^{n-2} = 13\cdot 3^{n-2} - 3\cdot 2^n.$$

Теорема доказана.

Теорема 2. При последовательном вычислении свертки сложность по числу операций сложения (вычитания) равна $10\cdot 3^{n-2} + (n-2)\cdot 2^n$ для нахождения $R_{4\cdot 3^{n-2}}$.

Доказательство. Особенностью вычисления $R_{4\cdot 3^{n-2}}$ является то, что нет необходимости вычислять БПУ меньших сверток в основных секциях по аналогии с X. Вычисляется БПУ только начальной основной секции $((R_{N=2^n})_0)$, используя свойство сложения результатов БПУ векторов длин M и $M/2$, доказанное в Лемме 2. Значения секций меньшей длины $M/2$ добавляются последовательно к секциям длины M таким образом, что после вычисления БПУ начальной основной секции $((R_{N=2^n})_0)$ результат начальной секции будет содержать корректировку, как если бы БПУ меньших секций были вычислены и добавлены к секциям большей длины.

Общее число операций сложения (вычитания) на Шаге 9б (табл. 4), необходимых для подготовки вычисления БПУ каждой секции, может быть получено как сумма геометрической убывающей прогрессии по следующей формуле:

$$Q^{\pm}_{Шаг\ 9б}(R_T)_i = 8\cdot 3^{n-3} \cdot \left(\frac{1-q^{i+1}}{1-q}\right),\ T = 4\cdot 3^{n-2} \cdot q^{i+1},\ i = \overline{0, n-3},\ q = \frac{2}{3},$$

где $Q^{\pm}_{Шаг\ 9б}(R_T)_i$ – сумма числа операций сложения (вычитания) над элементами R с итерации 0 по i включительно.

T отражает число элементов секций, которые были откорректированы значениями меньших секций на итерации i Шага 9б.

Таблица 4 – Итерационная обработка $R_{4\cdot 3^{n-2}}$ (Шаг 9б) перед вычислением ОБПУ при нахождении сверток длины $N = 2^n$, $n = 3,4,5,6$ (число операций сложения и вычитания)

i	$N = 8, n = 3$
0	+4,–4.[7]
i	$N = 16, n = 4$
0	+4,–4, +4,–4, +4,–4.[7]
1	+8,–8.[7]
i	$N = 32, n = 5$
0	+4,–4, +4,–4, +4,–4, +4,–4, +4,–4, +4,–4, +4,–4, +4,–4, +4,–4.[7]
1	+8,–8, +8,–8, +8,–8.[7]
2	+16,–16.[7]

i	$N = 64, n = 6$
0	+4,–4, +4,–4.[7]
1	+8,–8, +8,–8, +8,–8, +8,–8, +8,–8, +8,–8, +8,–8, +8,–8, +8,–8.[7]
2	+16,–16, +16,–16, +16,–16.[7]
3	+32, –32.[7]

Примечание 7. Пробелы добавлены для удобства восприятия. Знаки «+» и «–» обозначают число операций сложения и вычитания соответственно по секциям.

Общее число однословных операций сложения (вычитания) над элементами R на Шаге 9б по завершении итерации $i = n-3$ может быть выражено формулой $Q^{\pm}_{Шаг\ 9б}(R) = 8\cdot 3^{n-2} - 2^{n+1}$ при $n \ge 2$.

С учетом 3^{n-2} операций сложения, 3^{n-2} операций вычитания на Шаге 8 и $n\cdot 2^n$ операций сложения (вычитания) на Шаге 10 для вычисления ОБПУ общее число операций сложения (вычитания), необходимых для вычисления R, примет вид:

$$Q^{\pm}(R_{4\cdot 3^{n-2}}) = Q^{\pm}_{Шаг\ 8}(R_{4\cdot 3^{n-2}}) + Q^{\pm}_{Шаг\ 9б}(R_{4\cdot 3^{n-2}}) + Q^{\pm}_{Шаг\ 10}((R_{2^n})_0).$$

$$Q^{\pm}(R_{4\cdot 3^{n-2}}) = \left(2\cdot 3^{n-2}\right) + \left(8\cdot 3^{n-2} - 2^{n+1}\right) + \left(n\cdot 2^n\right) = 10\cdot 3^{n-2} + (n-2)\cdot 2^n.$$

Теорема доказана.

Теорема 3. При последовательном вычислении свертки общая сложность по числу однословных операций сложения (вычитания) равна $28\cdot 3^{n-2} + (n-3)\cdot 2^{n+1}$.

Доказательство. Общее число операций сложения (вычитания) имеет вид:

$$Q^{\pm} = Q^{\pm}_X + Q^{\pm}_Y + Q^{\pm}_R,$$

где $Q^{\pm}_X$, $Q^{\pm}_Y$, $Q^{\pm}_R$ – число операций сложения (вычитания) при обработке X, Y и R.

При подстановке значений, полученных в Лемме 1 и Теоремах 2, 3, приходим к следующему выражению:

$$Q^{\pm} = \left(5\cdot 3^{n-2} + (n-1)\cdot 2^n\right) + \left(13\cdot 3^{n-2} - 3\cdot 2^n\right) + \left(10\cdot 3^{n-2} + (n-2)\cdot 2^n\right).$$

После суммирования приходим к искомому выражению. Теорема доказана.

Число операций при параллельном вычислении алгоритма. На данное время частота процессора достигает своего физического предела. В то же самое время требования по времени выполнения операций над большими числами усиливаются. Кластерные решения позволяют достигать требуемых условий по быстродействию, но являются дорогостоящими, что не дает возможности широкого применения. Одним из решений является интегрирование большого числа ядер на одном кристалле, что удешевляет аппаратную часть решения. Такие системы используют потоковые процессоры, особен-

ностью которых является параллельное выполнение однотипной операции над большим массивом данных. Примером таких плат можно привести графические ускорители компаний NVIDIA и AMD (последние модели: GeForce GTX 580 с 512 потоковыми процессорами, Radeon HD 6990 с 3072 потоковыми процессорами). Изначально графические ускорители (GPU – Graphic Processor Unit) разрабатывались для решения задач быстрой обработки графической информации, которые могли быстро выполнять параллельно небольшой набор арифметических операций над векторами и массивами. Со временем графические ускорители переросли задачи графической обработки, набор команд вырос до такой степени, что ускорители стали называть GPGPU (General Purpose Graphic Processor Unit – графический ускоритель общего назначения). Методы, эффективные при последовательном выполнении, не эффективно использовать при реализации на параллельных (потоковых) процессорах графических ускорителей. Последовательные методы используют большой набор арифметических операций и функций на разных шагах алгоритма, что делает их сильно зависимыми друг от друга и требует ожидания выполнения текущего шага перед тем, как перейти к вычислению следующего шага, что ограничивает число одновременно задействованных потоковых процессоров. Для использования графических ускорителей необходимы новые алгоритмы, которые распределяют вычисления таким образом, чтобы задействовать максимально свободные процессоры, что в свою очередь уменьшает общее число операций, выполненных каждым из процессоров, и значительно уменьшает общее время выполнения операции, реализуемой данным алгоритмом. При разработке таких алгоритмов необходимо находить баланс между разбивкой на число таких шагов, каждый из которых имеет однотипные независимые друг от друга операции, и числом операций на каждом шаге.

Особенностью представленного алгоритма является то, что все операции в итерации по j являются независимыми. Если число параллельных (потоковых) процессоров превышает число операций алгоритма, необходимых для завершения итерации i, то достаточно одной операции (такта) потоковых процессоров для выполнения итерации i. Если шаг алгоритма состоит из нескольких итераций по i (например $n-2$ итерации) и число задействованных потоковых процессоров превышает (или равно) число операций на каждой итерации i, то время выполнения данного шага будет равно времени выполнения $n-2$ операций потоковых процессоров с учетом того, что все процессоры работают параллельно. Продолжая данные рассуждения, рассмотрим сложность по числу операций сложения (вычитания) и умножения каждого шага предложенного алгоритма при параллельном выполнении в следующей теореме. Для удобства будем считать, что освободившиеся процессоры не начинают вычислений по итерации $i+1$ до тех пор, пока не выполнится последняя операция на итерации i. Перед нахождением оценок сложности при параллельной модели вычислений рассмотрим следующую лемму.

Лемма 5. Сложность по числу операций при параллельном вычислении I итераций на P ($I << P$) процессорах может быть получена на основе сложности по числу операций при последовательном вычислении с учетом того, что процессоры не начинают новую итерацию, пока не закончена текущая итерация, используя следующее выражение:

$$Q_{парал.вычисление} \le \left\lfloor \frac{Q_{посл.вычисление}}{P} \right\rfloor + I ,$$

где $Q_{посл.вычисление} = \sum_{i=0}^{I-1} Q_{i,\, посл.вычисление}$ – общее число операций при последовательном вычислении I итераций, каждая их которых поддается распараллеливанию.

Доказательство. Максимальное число операций, выполненное одним из параллельных P процессоров, на I итерациях может быть представлено в следующем виде:

$$Q_{посл.вычисление} = \sum_{i=0}^{I-1} \left\lceil \frac{Q_{i,\, посл.вычисление}}{P} \right\rceil.$$

Изменим предыдущую формулу таким образом, чтобы использовать округление к меньшему числу вместо округления к большему числу, что позволит избежать использования знака суммы:

$$Q_{посл.вычисление} = \sum_{i=0}^{I-1} \left\lfloor \frac{Q_{i,\, посл.вычисление} + P - 1}{P} \right\rfloor \geq \left\lfloor \frac{Q_{посл.вычисление}}{P} \right\rfloor + \left\lceil \frac{I \cdot (P-1)}{P} \right\rceil.$$

Если $I << P$, то приходим к искомому выражению. Лемма доказана.

Теорема 4. При параллельном вычислении свертки $R_N \leftarrow X_N \otimes Y_N$ длины $N = 2^n$ на P процессорах сложность по числу операций однословного сложения, вычитания и умножения равна $Q^{\pm,*} \leq \left\lfloor \frac{33 \cdot 3^{n-2} + (n-3) \cdot 2^{n+1}}{P} \right\rfloor + 6n - 4$.

Доказательство. Число операций при параллельном вычислении можно представить в виде:

$$Q^{\pm}_{Шаг\,1,\,2\,и\,5}(X, XADD) = Q^{\pm}_{Шаг\,1}(X) + Q^{\pm}_{Шаг\,2}(X) + Q^{\pm}_{Шаг\,5}(XADD),$$

$$Q^{\pm}_{Шаг\,3б,\,3в\,и\,4}(Y, YADD) = Q^{\pm}_{Шаг\,3б}(Y) + Q^{\pm}_{Шаг\,3в}(Y) + Q^{\pm}_{Шаг\,4}(YADD),$$

$$Q^{\pm}_{Шаг\,8,\,9б\,и\,10}(R) = Q^{\pm}_{Шаг\,8}(R) + Q^{\pm}_{Шаг\,9б}(R) + Q^{\pm}_{Шаг\,10}(R),$$

где $Q^{\pm}_{Шаг\,1}(X)$, $Q^{\pm}_{Шаг\,2}(X)$, $Q^{\pm}_{Шаг\,5}(XADD)$ – число операций на Шагах 1,2 и 5 для X, $XADD$; $Q^{\pm}_{Шаг\,3б}(Y)$, $Q^{\pm}_{Шаг\,3в}(Y)$, $Q^{\pm}_{Шаг\,4}(YADD)$ – число операций на Шагах 3б, 3в и 4 для Y, $YADD$; $Q^{\pm}_{Шаг\,8}(R)$, $Q^{\pm}_{Шаг\,9б}(R)$, $Q^{\pm}_{Шаг\,10}(R)$ – число операций на Шагах 8, 9б и 10 для R.

На Шаге 3б на одной итерации по j элементы добавляются к Y на основе элементов секции $((Y_M)_j)$, которая модифицируется на Шаге 3в, поэтому Шаги 3б и 3в являются зависимыми. При параллельном вычислении Шагов 3б и 3в нельзя гарантировать, что элементы добавятся к Y до того, как секция будет модифицирована. Поэтому считаем, что Шаги 3в и 3б выполняются последовательно и представляют отдельные итерации. Выражения $r_{4j+1} \leftarrow r_{4j+1} - radd_j$, $r_{4j+3} \leftarrow r_{4j+3} + radd_j$ на Шаге 8 не зависят друг от друга, поэтому они могут быть вычислены в одной итерации.

Таблица 5 – Число операций и итераций на Шагах 1, 2, 5, 3б, 3в, 4, 8, 9б, 10 при последовательном вычислении $X, XADD$, $Y, YADD$, R

$X, XADD$			$Y, YADD$			R		
Шаг	Итераций	Операций	Шаг	Итераций	Операций	Шаг	Итераций	Операций
1	n	$n \cdot 2^n$	3б	$n-2$	$4 \cdot 3^{n-2} - 2^n$	8	1	$2 \cdot 3^{n-2}$
2	$n-2$	$4 \cdot 3^{n-2} - 2^n$	3в	$n-2$	$8 \cdot 3^{n-2} - 2^{n+1}$	9б	$n-2$	$8 \cdot 3^{n-2} - 2^{n+1}$
5	1	3^{n-2}	4	1	3^{n-2}	10	n	$n \cdot 2^n$

С учетом Леммы 5 и табл. 5 оценки для параллельной модели вычислений можно представить в виде:

$$Q^{\pm}_{\text{Шаг 1, 2 и 5}}(X, XADD) \le \left(\left\lfloor \frac{n \cdot 2^n}{P} \right\rfloor + n\right) + \left(\left\lfloor \frac{4 \cdot 3^{n-2} - 2^n}{P} \right\rfloor + n - 2\right) + \left(\left\lfloor \frac{3^{n-2}}{P} \right\rfloor + 1\right);$$

$$Q^{\pm}_{\text{Шаг 3б, 3в и 4}}(Y, YADD) \le \left(\left\lfloor \frac{4 \cdot 3^{n-2} - 2^n}{P} \right\rfloor + n - 2\right) + \left(\left\lfloor \frac{8 \cdot 3^{n-2} - 2^{n+1}}{P} \right\rfloor + n - 2\right) + \left(\left\lfloor \frac{3^{n-2}}{P} \right\rfloor + 1\right);$$

$$Q^{\pm}_{\text{Шаг 8, 9б и 10}}(R) \le \left(\left\lfloor \frac{2 \cdot 3^{n-2}}{P} \right\rfloor + 1\right) + \left(\left\lfloor \frac{8 \cdot 3^{n-2} - 2^{n+1}}{P} \right\rfloor + n - 2\right) + \left(\left\lfloor \frac{n \cdot 2^n}{P} \right\rfloor + n\right);$$

$$Q^{*} \le \left\lfloor \frac{5 \cdot 3^{n-2}}{P} \right\rfloor + 1.$$

После группировки приходим к выражениям:

$$Q^{\pm}_{\text{Шаг 1, 2 и 5}}(X, XADD) \le \left\lfloor \frac{5 \cdot 3^{n-2} + (n-1) \cdot 2^n}{P} \right\rfloor + 2n - 1,$$

$$Q^{\pm}_{\text{Шаг 3б,3в и 4}}(Y, YADD) \le \left\lfloor \frac{13 \cdot 3^{n-2} - 3 \cdot 2^n}{P} \right\rfloor + 2n - 3;$$

$$Q^{\pm}_{\text{Шаг 8, 9б и 10}}(R) \le \left\lfloor \frac{10 \cdot 3^{n-2} + (n-2) \cdot 2^n}{P} \right\rfloor + 2n - 1; \; Q^{*} \le \left\lfloor \frac{5 \cdot 3^{n-2}}{P} \right\rfloor + 1.$$

После суммирования всех выражений приходим к искомому выражению. Теорема доказана.

Тестовый пример вычисления свертки длины $N = 16$. В табл. 6 считается, что индексирование в векторах начинается с нуля и элемент с нулевым индексом расположен слева. Длина одного разряда равна $\omega = 32$ бит.

Таблица 6 – Пример вычисления свертки предложенным алгоритмом при $N = 16$, $n = 4$

Шаг №	i, j	Значение вектора после Шага № на итерации по i, j
Шаг 0		X_{16} =1,2,3,4,5,6,7,8,0,0,0,0,0,0,0,0.^{8} Y_{16} =0,0,0,0,0,0,0,0,8,7,6,5,4,3,2,1.^{8}
Шаг 1		$((X_{16})_0)$ =36,36,16,¯16,¯8,¯8,0,0,¯4,¯4,0,0,0,0,0,0
Шаг 2	$i = 0$, $j = 0$	X_{24} =36,36,¯16,¯16,¯8,¯8,0,0,¯4,¯4,0,0,0,0,0,0,40,40,¯16,¯16,¯8,¯8,0,0
Шаг 2	$i = 1$, $j = 0$	X_{28} =36,36,¯16,¯16,¯8,¯8,0,0,¯4,¯4,0,0,0,0,0,0,40,40,¯16,¯16,¯8,¯8,0,0, 44,44,¯16,¯16
Шаг 2	$i = 1$, $j = 1$	X_{32} =36,36,¯16,¯16,¯8,¯8,0,0,¯4,¯4,0,0,0,0,0,0,40,40,¯16,¯16,¯8,¯8,0,0, 44,44,¯16,¯16,¯4,¯4,0,0
Шаг 2	$i = 1$, $j = 2$	X_{36} =36,36,¯16,¯16,¯8,¯8,0,0,¯4,¯4,0,0,0,0,0,0,40,40,¯16,¯16,¯8,¯8,0,0, 44,44,¯16,¯16,¯4,¯4,0,0,48,48,¯16,¯16
Шаг 3б	$i = 0$, $j = 0$	Y_{24} =0,0,0,0,0,0,0,0,8,7,6,5,4,3,2,1,0,0,0,8,¯2,¯2,¯2,¯2
Шаг 3в	$i = 0$, $j = 0$	Y_{24} =0,0,0,0,15,11,7,3,0,0,0,0,1,1,1,1,0,0,0,8,¯2,¯2,¯2,¯2

Продолж. табл. 6

Шаг 3б	$i=1,\ j=0$	Y_{28} =0,0,0,0,15,11,7,3,0,0,0,0,1,1,1,1,0,0,0,8,¯2,¯2,¯2,¯2,0,15,¯8,¯7
Шаг 3в	$i=1,\ j=0$	Y_{28} =0,0,26,10,0,0,4,4,0,0,0,0,1,1,1,1,0,0,0,8,¯2,¯2,¯2,¯2,0,15,¯8,¯7
Шаг 3б	$i=1,\ j=1$	Y_{32} =0,0,26,10,0,0,4,4,0,0,0,0,1,1,1,1,0,0,0,8,¯2,¯2,¯2,¯2,0,15,¯8,¯7,0,1,0,¯1
Шаг 3в	$i=1,\ j=1$	Y_{32} =0,0,26,10,0,0,4,4,0,0,2,2,0,0,0,0,0,0,0,8,¯2,¯2,¯2,¯2,0,15,¯8,¯7,0,1,0,¯1
Шаг 3б	$i=1,\ j=2$	Y_{36} =0,0,26,10,0,0,4,4,0,0,2,2,0,0,0,0,0,0,0,8,¯2,¯2,¯2,¯2,0,15,¯8,¯7,0,1,0,¯1, 0,¯2,0,2
Шаг 3в	$i=1,\ j=2$	Y_{36} =0,0,26,10,0,0,4,4,0,0,2,2,0,0,0,0,0,8,¯4,¯4,0,¯8,0,0,0,15,¯8,¯7,0,1,0,¯1, 0,¯2,0,2
Шаг 4		Y_{36} =36,¯36,16,¯16,8,¯8,0,0,4,¯4,0,0,0,0,0,0,0,16,¯8,¯8,¯8,¯8,8,8, 0,30,¯16,¯14,0,2,0,¯2,0,¯4,0,4
Шаг 5		$XADD_9$ =52,¯8,¯4,0,56,¯8,60,¯4,64 $YADD_9$ =¯52,¯8,¯4,0,8,0,16,0,0
Шаг 6		R_{36} =2592,¯2592,¯512,512,¯128,128,0,0,¯32,32,0,0,0,0,0,0,0,0,1280,256,256, 128,128,0,0,0,2640,512,448,0,¯16,0,0,0,¯384,0,¯128
Шаг 7		$RADD_9$ =¯2704,64,16,0,448,0,960,0,0
Шаг 8		R_{36} =2592,112,¯512,¯2192,¯128,64,0,64,¯32,16,0,16,0,0,0,0,0,832,256,704, 128,128,0,0,0,1680,512,1408,0,¯16,0,0,0,¯384,0,¯128
Шаг 9б	$i=0$	R_{24} =2592,952,¯256,¯1488,¯128,¯776,¯256,¯640,¯32,8,0,16,0,8,0,0, 0,640,256,640,128,320,0,64
Шаг 9б	$i=1$	R_{16} =2592,1272,¯128,¯1168,¯64,¯616,¯256,¯608,¯32,¯312,¯128,¯304,¯64, ¯152,0,¯32
Шаг 10		R_{16} =0,64,112,145,164,170,164,147,120,84,56,35,20,10,4,1.[8]

Примечание 8. Результат вычисления свертки, взятый в обратном порядке, равен произведению двух восьмиразрядных чисел, равных (1,2,3,4,5,6,7,8). Входные параметры длины $N/2$ могут быть преобразованы несложным образом так, что результат циклической свертки длины N будет равен произведению двух восьмиразрядных чисел [3]. Программная реализация (xx3fwalsh6u5g) алгоритма вычисления циклической свертки на языке программирования APL занимает одну страницу.

Вывод

Рассмотрен новый алгоритм вычисления циклической свертки длины $N=2^n$ на основе быстрого преобразования Уолша (БПУ) для параллельной модели вычислений (SIMD – Single Instruction Multiple Data). Показано, что данный алгоритм позволяет уменьшить более чем в два раза число вычислений БПУ, что значительно уменьшает общую сложность по числу операций сложения и вычитания по сравнению с [3]. В новом алгоритме применены следующие резервы оптимизации: БПУ в дополнительных секциях не вычисляются, в результирующих секциях вычисляется только одно ОБПУ длины $N=2^n$ (за счет оригинального использования свойства корректировки сигнала длины M значениями сигнала длины $M/2$ перед вычислением ОБПУ большего сигнала). Метод проиллюстрирован на реализации сверток длины 4 и 8 в общем виде, приведена схема вычисления свертки длины 8. Приведен тестовый пошаговый пример для вычисления циклической свертки длины $N=16$ предложенным методом. В виде лемм и теорем приведены оценки сложности по числу операций при параллельном и последовательном вычислении.

Литература

1. David A. Pitassi. Fast convolution using the Walsh transform / David A. Pitassi // Applications of Walsh Functions, Proceeding. – 1971. – April. – P. 130-133.
2. Davis W.F. A class of efficient convolution algorithms / W.F. Davis // Applicat. Walsh Functions. – 1972. – March. – P. 318-329.
3. Реализация операции умножения с использованием преобразования Уолша / А.Н. Терещенко, С.С. Мельникова, Л.А. Гнатив [и др.] // Проблемы управления и информатики. – 2010. – № 2. – С. 102-126.
4. Терещенко А.Н. Оптимизация метода Питасси вычисления свертки / А.Н. Терещенко // Искусственный интеллект. – 2009. – № 1. – С. 204-212.
5. Задирака В.К. Анализ сложности алгоритма умножения сверхбольших чисел на основе коэффициентов Уолша / В.К. Задирака, С.С. Мельникова // Кибернетика и системный анализ. – 2001. – № 6. – С. 99-110.
6. Гнатив Л.А. Эффективные алгоритмы быстрых преобразований Уолша с различными системами упорядочивания / Л.А. Гнатив, В.Е. Коссов // Методы и средства обработки информации в системах реального времени. – Киев : Ин-т кибернетики им. В.М. Глушкова АН УССР, 1990. – С. 43-49.

Literatura

1. David A. Applications of Walsh Functions. 1971. Proceeding. P.130-133. April. 1971.
2. Davis W. F. Applicat. Walsh Functions.1972. March. P.318-329.
3. Tereshhenko A. N. Mezhdunarodnyj nauchno-tehnicheskij zhurnal Problemy upravlenija i informatiki. 2010. № 2. S. 102-126.
4. Tereshhenko A. N. Iskusstvennyj intellekt. 2009. №1. S. 204-212.
5. Zadiraka V. K. Kibernetika i sistemnyj analiz. 2001. № 6. S. 99-110.
6. Gnativ L.A. Metody i sredstva obrabotki informacii v sistemah real'nogo vremeni. Kiev: In-t kibernetiki im. V.M. Glushkova AN USSR. 1990. S. 43-49.

RESUME

A.N. Tereshchenko, V.K. Zadiraka

Parallel Computation Algorithm for Convolution

There is a new computation algorithm for cyclic convolution of length $N=2^n$ based on Fast Walsh transformation (FWT) in parallel model (SIMD – Single Instruction Multiple Data). The total number of FWTs in new algorithm is reduced more than twofold that decreases the number of single precision additions and subtractions on sequential computation. There are several applied optimizations: FWTs are not computed in additional sections, there is only one FWT of length $N=2^n$ in resulting sections. The last optimization uses the feature of pre-computational adjustment of multi-digit value of length M by multi-digit value of length $M/2$ to get the sum of FWTs both multi-digit values based on computation of one FWT multi-digit value of length M .

In the article, the new method of several FWTs computation in parallel is described. The computation scheme for convolution of length 8 is shown. The operator W on the scheme is applied for both FWTs of length 4 and 8 at the same time on parallel computation.

The method of getting complexity of parallel computation based on complexity of consequential computation is described. With use of this method, complexity of sequential and parallel computations is shown.

Статья поступила в редакцию 31.05.2012.